IT421-Web Site Development
INF414-Web Based Information Systems

Dr. Islam Taj-Eddin

IT Dept., FCI, Assiut Univ.

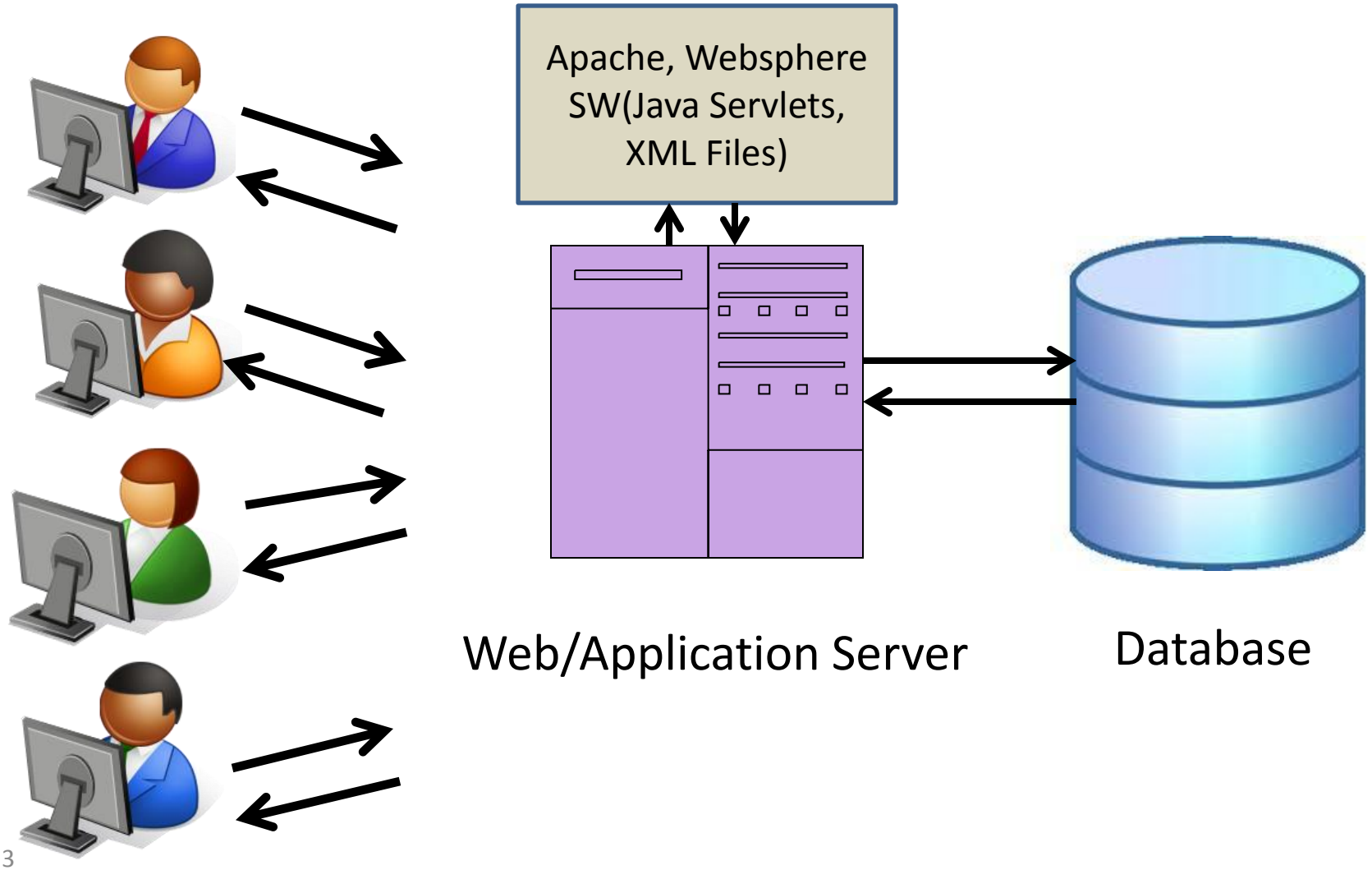
Introduction to PHP

Server side basics

http://server/path/file

- Usually when you type a URL in your browser:
  - Your computer looks up the server's IP address using DNS
  - Your browser connects to that IP address and requests the given file
  - The web server software (e.g. Apache) grabs that file from the server's local file system
  - The server sends back its contents to you

Server side basics



Server side basics

http://**www.facebook.com/home.php**

- Some URLs actually specify programs that the web server should *run*, and then send their output back to you as the result:
  - The above URL tells the server **facebook.com** to run the program **home.php** and send back its output

Server-Side web programming

- Server-side pages are programs written using one of many web programming languages/frameworks
  - examples: PHP, Java/JSP, Ruby on Rails, ASP.NET, Python, Perl

Server-Side web programming

- Also called *server side scripting*:
  - Dynamically edit, change or add any content to a Web page
  - Respond to user queries or data submitted from HTML forms
  - Access any data or databases and return the results to a browser
  - Customize a Web page to make it more useful for individual users
  - Provide security since your server code cannot be viewed from a browser

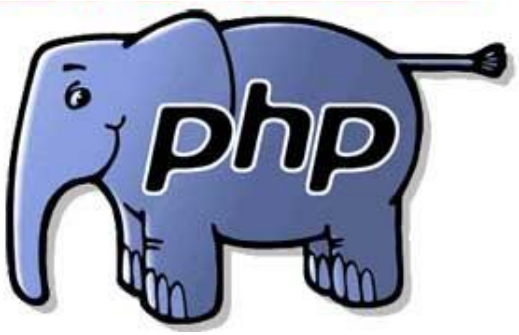
Server-Side web programming

- Web server:
  - contains software that allows it to run server side programs
  - sends back their output as responses to web requests

- Each language/framework has its pros and cons
  - we use PHP

What is PHP?

- PHP stands for "PHP Hypertext Preprocessor"
- Server-side scripting language
- Used to make web pages dynamic:
  - provide different content depending on context
  - interface with other services: database, e-mail, etc.
  - authenticate users
  - process form information
- PHP code can be embedded in XHTML code



Lifecycle of a PHP web request

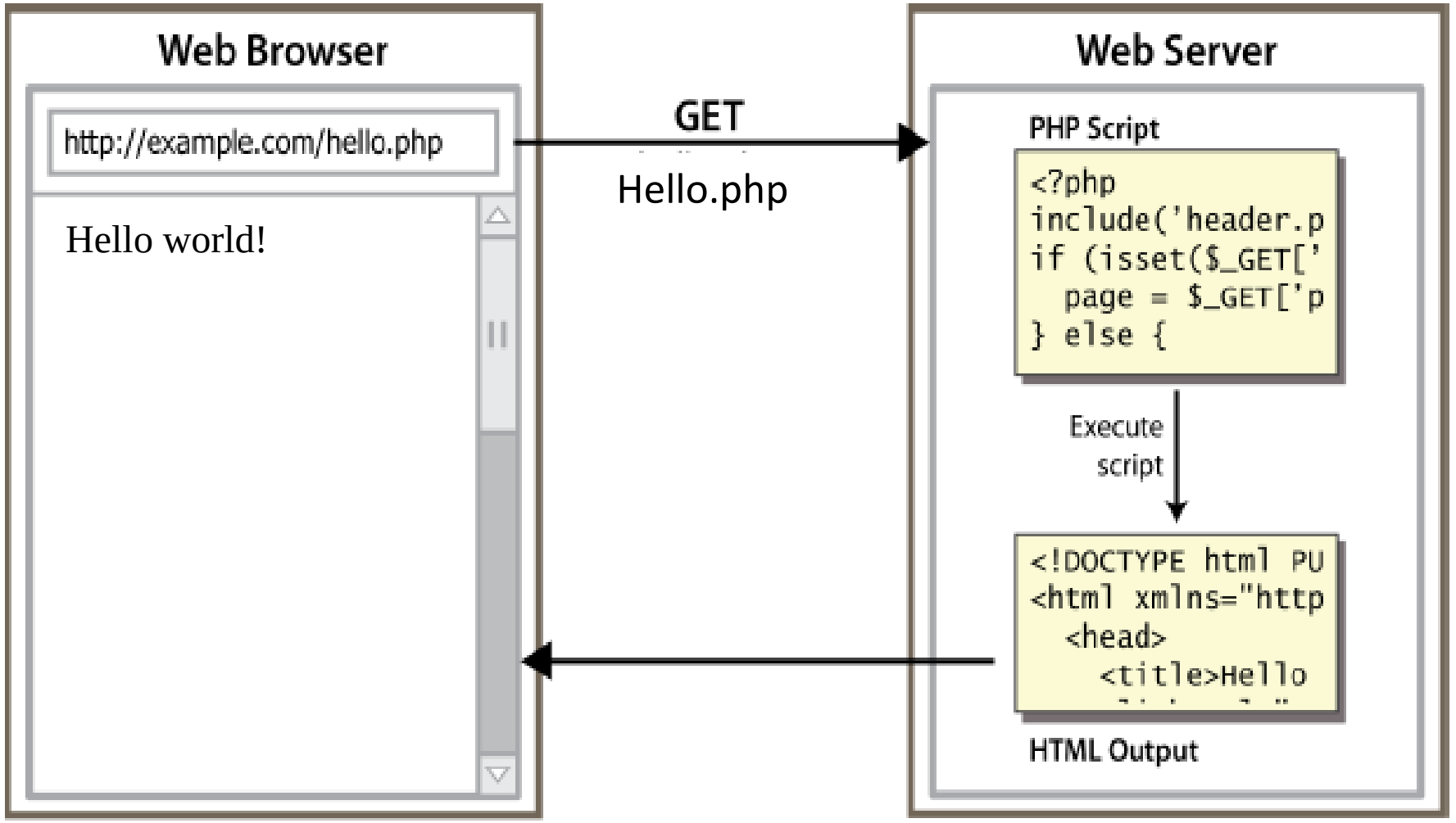
- browser requests a .html file (static content):
  - server just sends that file

- browser requests a .php file (dynamic content):
  - server reads it, runs any script code inside

- it, then sends result across the network

- script produces output that becomes the response sent back

Lifecycle of a PHP web request



User's computer

Server computer

Why PHP?

- Free and open source
- Compatible
  - as of November 2006, there were more than 19 million websites (domain names) using PHP.
- Simple
- Interpreted, not compiled.
- Procedural and OO.
- Not standalone.
- Relaxed syntax: fewer, looser data types

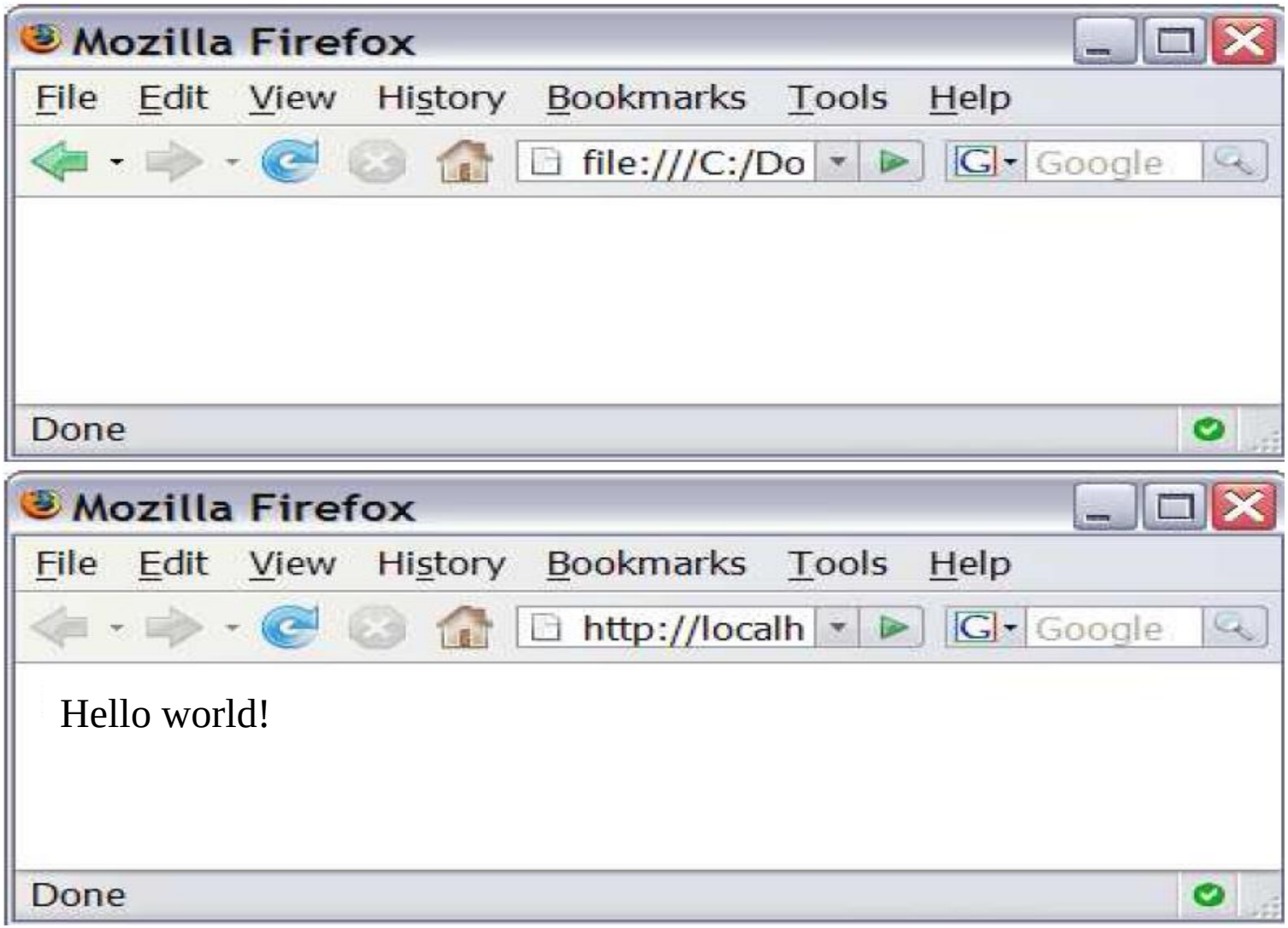
Hello World!

```php
<?php
print "Hello, world!";
?>
```
PHP

```
Hello world!
```
output

- you can't view your .php page on your local hard drive; you'll either see nothing or see the PHP source code
- if you upload the file to a PHP-enabled web server, requesting the .php file will run the program and send you back its output

Viewing PHP output



PHP syntax template

```
HTML content
<?php
PHP code
?>
HTML content
<?php
PHP code
?>
HTML content ...
```
PHP

- Contents of a .php file between `<?php` and `?>` are executed as PHP code
- All other contents are output as pure HTML
- We can switch back and forth between HTML and PHP "modes"

Console output: `print`

```
print "text";
```
PHP

```
print "Hello, World!\n";
print "Escape \"chars\" are the SAME as in Java!\n";
print "You can have line breaks in a string.";
print 'A string can use "single-quotes". It\'s cool!';
```
PHP

```
Hello world!  Escape "chars" are the SAME as in Java! You can have line
breaks in a string. A string can use "single-quotes". It's cool!
```
output

- some PHP programmers use the equivalent echo instead of print
- \n has no effect, no line change

Variables

```
$name = expression;
```
PHP

```
$user_name = "mundruid78";
$age = 16;
$drinking_age = $age + 5;
$this_class_rocks = TRUE;
```
PHP

- names are case sensitive
- names always begin with $, on both declaration and usage
- always implicitly declared by assignment (type is not written)
- a loosely typed language (like JavaScript or Python)

Variables

- basic types: *int, float, boolean, string, array, object, NULL*
  - test type of variable with `is_type` functions, e.g. `is_string`
  - `gettype` function returns a variable's type as a string
- PHP *converts between types automatically* in many cases:
  - string → int auto-conversion on +
  - int → float auto-conversion on /
- type-cast with **(type):**
  - `$age = (int) "21";`

Arithmetic operators

- + - * / % . ++ --
- = += -= *= /= %= .=
- many operators auto-convert types: 5 + "7" is 12

Comments

```
# single-line comment
// single-line comment
/*
multi-line comment
*/
```
PHP

- like Java, but # is also allowed
- a lot of PHP code uses # comments instead of //

`String` Type

```php
$favorite_food = "Egyptian";
print $favorite_food[2]; #y
$favorite_food = $favorite_food . " cuisine";
print $favorite_food;
```

PHP

- zero-based indexing using bracket notation
- there is no char type; each letter is itself a String
- string concatenation operator is . (period), not +
  - `5 + "2 turtle doves" === 7`
  - `5 . "2 turtle doves" === "52 turtle doves"`
- can be specified with "" or ''

`String` Functions

```php
$name = "Stefanie Hatcher";
$length = strlen($name);
$cmp = strcmp($name, "Brian Le");
$index = strpos($name, "e");
$first = substr($name, 9, 5);
$name = strtoupper($name);
```

PHP

`String` Functions

Name	Java Equivalent
strlen	length
strpos	indexOf
substr	substring
strtolower, strtoupper	toLowerCase, toUpperCase
trim	trim
explode, implode	split, join
strcmp	compareTo

Interpreted Strings

```php
$age = 16;
print "You are " . $age . " years old.\n";
print "You are $age years old.\n"; # You are 16 years old.
```
PHP

- strings inside " " are interpreted
  - variables that appear inside them will have their values inserted into the string
- strings inside ' ' are not interpreted:

```php
print 'You are $age years old.\n'; # You are $age years
old. \n
```
PHP

Interpreted Strings

```
print "Today is your $ageth birthday.\n"; # $ageth not
found
print "Today is your {$age}th birthday.\n";
```

PHP

- if necessary to avoid ambiguity, can enclose variable in {}

Interpreted Strings

```php
$name = “Xenia";
$name = NULL;
if (isset($name)) {
print "This line isn't going to be reached.\n";
}
```
PHP

- a variable is NULL if
  - it has not been set to any value (undefined variables)
  - it has been assigned the constant NULL
  - it has been deleted using the unset function
- can test if a variable is NULL using the isset function
- NULL prints as an empty string (no output)

`for` loop (same as Java)

```
for (initialization; condition; update) {
      statements;
}
```

PHP

```
for ($i = 0; $i < 10; $i++) {
      print "$i squared is " . $i * $i . ".\n";
}
```

PHP

`bool` (Boolean) type

```php
$feels_like_summer = FALSE;
$php_is_great = TRUE;

$student_count = 7;
$nonzero = (bool) $student_count; # TRUE
```

PHP

- the following values are considered to be FALSE (all others are TRUE):
  - 0 and 0.0 (but NOT 0.00 or 0.000)
  - "", "0", and NULL (includes unset variables)
  - arrays with 0 elements
- FALSE prints as an empty string (no output); TRUE prints as a 1

`if/else` statement

```
if (condition) {
      statements;
} elseif (condition) {
      statements;
} else {
      statements;
}
```

PHP

`while` loop (same as Java)

```
while (condition) {
      statements;
}
```
PHP

```
do {
      statements;
} while (condition);
```
PHP

- break and continue keywords also behave as in Java

Math operations

```php
$a = 3;
$b = 4;
$c = sqrt(pow($a, 2) + pow($b, 2));
```

PHP

math functions

abs	ceil	cos	floor	log	log10	max
min	pow	rand	round	sin	sqrt	tan

math constants

M_PI	M_E	M_LN2

`Int` and `Float` Types

```php
$a = 7 / 2; # float: 3.5
$b = (int) $a; # int: 3
$c = round($a); # float: 4.0
$d = "123"; # string: "123"
$e = (int) $d; # int: 123
```

PHP

- int for integers and float for reals
- division between two int values can produce a float

Arrays

```php
$name = array();          # create
$name = array(value0, value1, ..., valueN);
$name[index]              # get element value
$name[index] = value;     # set element value
$name[] = value;          # append
```

PHP

```php
$a = array();      # empty array (length 0)
$a[0] = 23;        # stores 23 at index 0 (length 1)
$a2 = array("some", "strings", "in", "an", "array");
$a2[] = "Ooh!";    # add string to end (at index 5)
```

PHP

- Append: use bracket notation without specifying an index
- Element type is not specified; can mix types

Array functions

function name(s)	description
count	number of elements in the array
print_r	print array's contents
array_pop, array_push, array_shift, array_unshift	using array as a stack/queue
in_array, array_search, array_reverse, sort, rsort, shuffle	searching and reordering
array_fill, array_merge, array_intersect, array_diff, array_slice, range	creating, filling, filtering
array_sum, array_product, array_unique, array_filter, array_reduce	processing elements

Array function example

```php
$tas = array("MD", "BH", "KK", "HM", "JP");
for ($i = 0; $i < count($tas); $i++) {
      $tas[$i] = strtolower($tas[$i]);
}
$morgan = array_shift($tas);
array_pop($tas);
array_push($tas, "ms");
array_reverse($tas);
sort($tas);
$best = array_slice($tas, 1, 2);
```

PHP

- the array in PHP replaces many other collections in Java
  - list, stack, queue, set, map, ...

`String` compare functions

Name	Function
strcmp	compareTo
strstr, strchr	find string/char within a string
strpos	find numerical position of string
str_replace, substr_replace	replace string

- Comparison can be:
  - Partial matches
  - Others
- Variations with non case sensitive functions
  - strcasecmp

`String` compare functions examples

```php
$offensive = array( offensive word1, offensive
word2);
$feedback = str_replace($offcolor, "%!@*",
$feedback);
```

PHP

```php
$test = "Hello World! \n";
print strpos($test, "o");
print strpos($test, "o", 5);
```

PHP

```php
$toaddress = "feedback@example.com";
if(strstr($feedback, "shop")
      $toaddress = "shop@example.com";
else if(strstr($feedback, "delivery")
      $toaddress = "fulfillment@example.com";
```

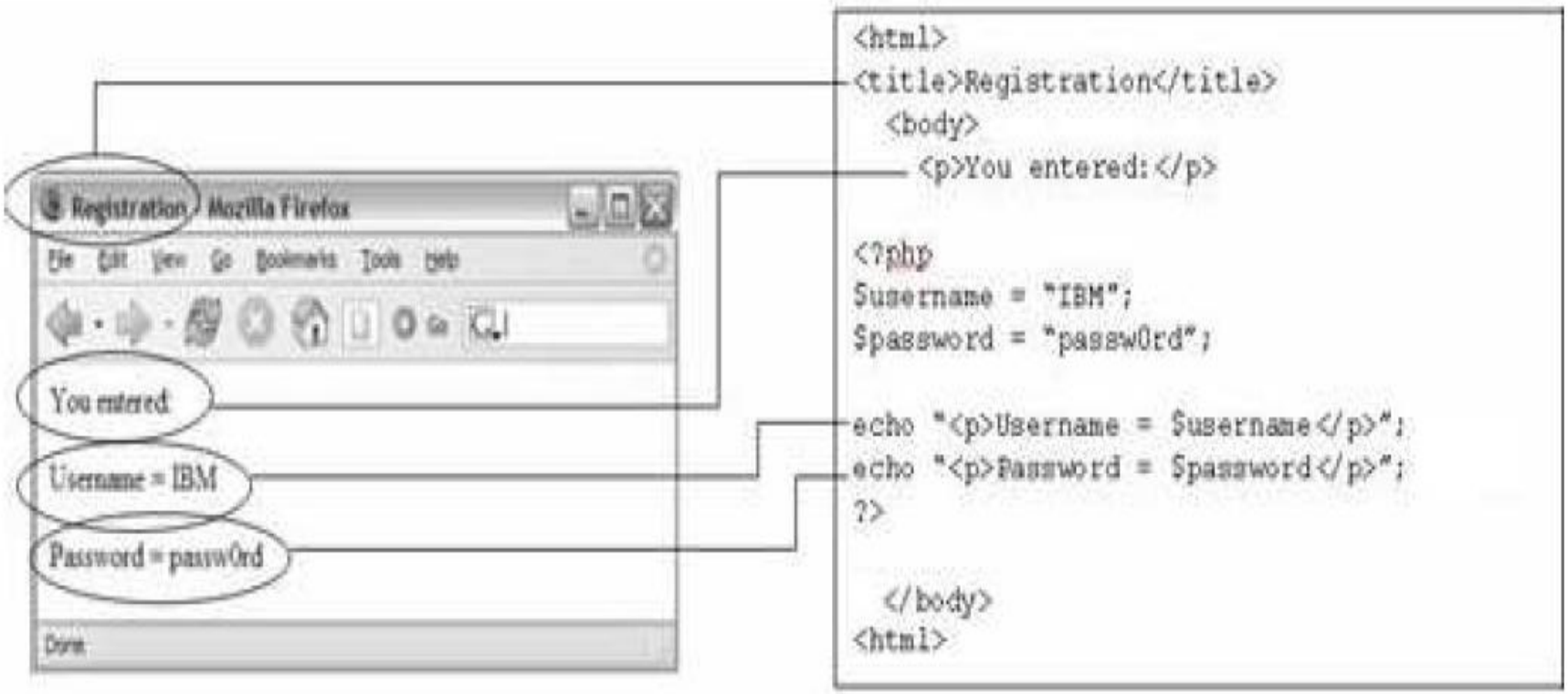
PHP

Regular expressions

```
[a-z]at                 #cat, rat, bat…
[aeiou]
[a-zA-Z]
[^a-z]                  #not a-z
[[:alnum:]]+            #at least one alphanumeric char
(very) *large           #large, very very very large…
(very){1, 3}                  #counting “very” up to 3
^bob                    #bob at the beginning
com$                    #com at the end                  PHPRegExp
```

- Regular expression: a pattern in a piece of text
- PHP has:
  - POSIX
  - **Perl regular expressions**

Embedding code in web pages



- most PHP programs actually produce HTML as their output
- dynamic pages; responses to HTML form submissions; etc.
- an embedded PHP program is a file that contains a mixture of HTML and PHP code

Printing HTML tags in PHP = bad style

```
<?php
print "<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.1//EN\"\n";
print " \"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd\">\n";
print "<html xmlns=\"http://www.w3.org/1999/xhtml\">\n";
print " <head>\n";
print " <title>My web page</title>\n";
...
 ?>
```

HTML

- best PHP style is to minimize print/echo statements in embedded PHP code
- but without print, how do we insert dynamic content into the page?
- don't print HTML; it's bad style!

PHP expression blocks

```
<?= expression ?>
```
PHP

```
<h2> The answer is <?= 6 * 7 ?> </h2>
```
PHP

```
The answer is 42
```
output

- PHP expression block: a small piece of PHP that evaluates and embeds an expression's value into HTML
  - `<?= expression ?>` is equivalent to:

```
<?php print expression; ?>
```
PHP

Expression block example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>CSE 190 M: Embedded PHP</title></head>
<body>
<?php
for ($i = 99; $i >= 1; $i--) {
?>
<p> <?= $i ?> first , <br />
<?= $i ?> second . <br />
third, <br />
<?= $i - 1 ?> first. </p>
<?php
}
?>
</body>
</html>
```

PHP

Common errors: unclosed braces, missing = sign

```
...
<body>
<p>Watch how high I can count:
<?php
for ($i = 1; $i <= 10; $i++) {
?>
      <? $i ?>
</p>
</body>
</html>
```

PHP

- if you forget to close your braces, you'll see an error about 'unexpected `$end`'
- if you forget = in `<?=`, the expression does not produce any output

Complex expression blocks

```
...
<body>
<?php
for ($i = 1; $i <= 3; $i++) {
      ?>
      <h<?= $i ?>>This is a level <?= $i ?>
                  heading.</h<?= $i ?>>
      <?php
}
?>
</body>
```

PHP

```
This is a level 1 heading.
This is a level 2 heading.
This is a level 3 heading.
```

output

Functions

```
function name(parameterName, ..., parameterName) {
   statements;
}
```
PHP

```
function quadratic($a, $b, $c) {
      return -$b + sqrt($b * $b - 4 * $a * $c) / (2
             * $a);
}
```
PHP

- parameter types and return types are not written
- a function with no return statements implicitly returns NULL

Default Parameter Values

```php
function print_separated($str, $separator = ", ") {
      if (strlen($str) > 0) {
            print $str[0];
            for ($i = 1; $i < strlen($str); $i++) {
                  print $separator . $str[$i];
            }
      }
}
```

PHP

```php
print_separated("hello"); # h, e, l, l, o
print_separated("hello", "-"); # h-e-l-l-o
```

PHP

- if no value is passed, the default will be used